

## Laboratorio Nro. 1

### Recursión

#### Objetivos:

1. Identificar el caso base y el caso general de un problema definido recursivamente
2. Usar ecuaciones de recurrencia para determinar la complejidad en tiempo y espacio y definidos de forma recursiva
3. Usar la notación  $O$  para encontrar formalmente la complejidad asintótica en espacio y memoria de algoritmos

#### Consideraciones iniciales

##### Leer la Guía



Antes de comenzar a resolver el presente laboratorio, leer la **“Guía Metodológica para la realización y entrega de laboratorios de Estructura de Datos y Algoritmos”** que les orientará sobre los requisitos de entrega para este y todos los laboratorios, las rúbricas de calificación, el desarrollo de procedimientos, entre otros aspectos importantes.

##### Registrar Reclamos



En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de <http://bit.ly/2q4TTKf>, el cual será atendido en la menor brevedad posible.

### Traducción de Ejercicios

En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea.



### Visualización de Calificaciones

A través de **Eafit Interactiva** encontrarán **un enlace** que les permitirá **ver un registro de las calificaciones** que **emite el docente** para cada taller de laboratorio y según las rubricas expuestas. **Véase sección 3, numeral 3.7.**



### GitHub

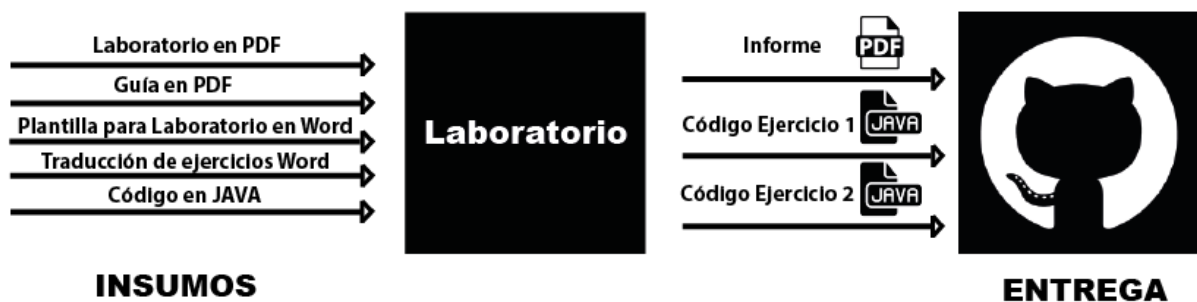


Crear un repositorio en su cuenta de GitHub con el nombre `st0245`. **2.** Crear una carpeta dentro de ese repositorio con el nombre `laboratorios`. **3.** Dentro de la carpeta `laboratorios`, crear una carpeta con nombre `lab01`. **4.** Dentro de la carpeta `lab01`, crear tres carpetas: `informe`, `codigo` y `ejercicioEnLinea`. **5.** Subir el informe pdf a la carpeta `infome`, el código del ejercicio 1 a la carpeta `codigo` y el código del ejercicio en línea a la carpeta `ejercicioEnLinea`. Así:

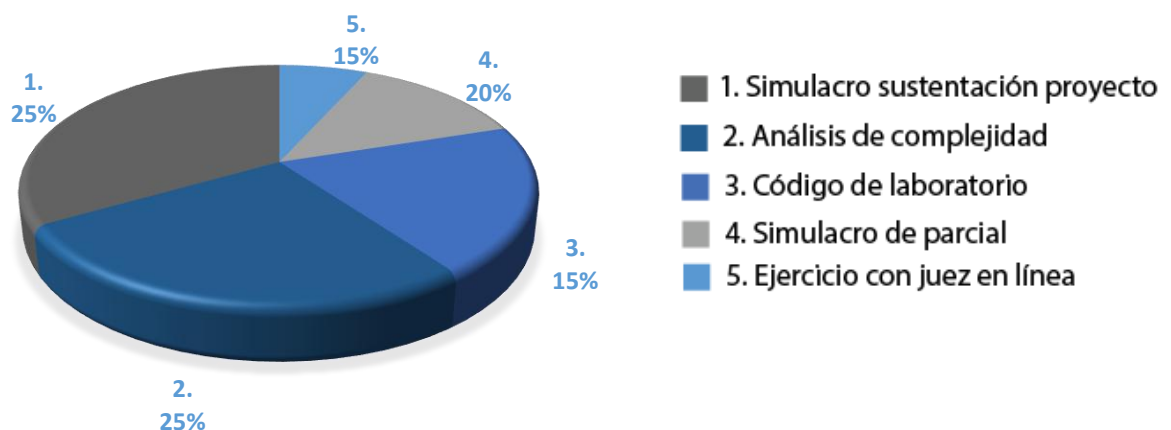
```
st0245
  laboratorios
    lab01
      informe
      codigo
      ejercicioEnLinea
    lab02
    ...
```

## Intercambio de archivos

Los archivos que **ustedes deben entregar** al docente son: **un archivo PDF** con el informe de laboratorio usando la plantilla definida, y **dos códigos**, uno con la solución al numeral 1 y otro al numeral 2 del presente. Todo lo anterior se entrega en **GitHub**.



## Porcentajes y criterios de evaluación para el laboratorio



DOCENTE MAURICIO TORO BERMÚDEZ  
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627  
Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)

## Ejercicios a resolver



En la vida real, la documentación de software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126



Véase Guía *en Sección 3, numeral 3.4*



Código de laboratorio en *GitHub*. Véase Guía en *Sección 4, numeral 4.24*









*Es opcional entregar documentación. Si lo hace, utilice **javadoc** o equivalente. No suba el HTML a GitHub.*



*No se reciben archivos en **.RAR** ni en **.ZIP***

**1.1** Escriban una función que reciba como parámetro un valor entero  $n$  y retorne el enésimo término de la serie de Fibonacci. Probar la función con valores de 10, 20, con 100 y con 1000.

## 2) Ejercicios en línea sin documentación HTML en GitHub

	Véase Guía en <b>Sección 3, numeral 3.3</b>		No se requiere documentación para los ejercicios en línea
	Entregar un archivo en <b>.JAVA</b>		No se reciben archivos en <b>.PDF</b> ni <b>.TXT</b>
	<b>Resolver</b> los problemas de <b>CodingBat</b> usando <b>Recursión</b>		Código del ejercicio en línea en <b>GitHub</b> . Véase Guía en <b>Sección 4, numeral 4.24</b>



**NOTA:** Recuerden que, si toman la respuesta de alguna fuente, deben referenciar según el tipo de cita correspondiente. Véase Guía en **Sección 4, numerales 4.16 y 4.17**

**2.1** Resolver al menos 5 ejercicios del nivel *Recursion 1* de CodingBat:  
<http://codingbat.com/java/Recursion-1>

**2.2** Resolver al menos 5 ejercicios del nivel *Recursion 2* de la página CodingBat:  
<http://codingbat.com/java/Recursion-2>



**NOTA:** No está permitido el ejercicio **GroupSum**.

**2.3 [Ejercicio Opcional]:** Resuelvan el siguiente problema  
<http://bit.ly/2k8CGSG>

**2.4 [Ejercicio Opcional]** Resuelvan el siguiente problema <http://bit.ly/2gTLZ53>

2.5 [Ejercicio opcional] Resuelvan el siguiente ejercicio <http://bit.ly/2hGqJPB>

2.6 [Ejercicio Opcional]: Resuelvan el siguiente ejercicio <http://bit.ly/2hrrCfS>

2.7 [Ejercicio Opcional]: Resuelvan el siguiente ejercicio <http://bit.ly/2k8CGSG>

### 3) Simulacro de preguntas de sustentación de Proyectos



Véase Guía en **Sección 3,**  
**Numeral 3.4**



Entregar informe de  
laboratorio en **PDF**



Usen la **plantilla** para  
responder laboratorios



**No apliquen Normas**  
**Icontec** para esto

3.1 Expliquen con sus propias palabras cómo funciona el ejercicio **GroupSum5**



**NOTA:** Recuerden que deben explicar su implementación en el informe PDF

3.2 Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y 2.2 y agréguenla al informe PDF

3.3 Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del ejercicio anterior

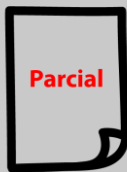
3.4 ¿Qué aprendieron sobre **Stack Overflow**? ¿Por qué sucede este error?

**3.5** ¿Cuál es el valor más grande que pudo calcular para Fibonacci? ¿Por qué? ¿Por qué no se puede ejecutar Fibonacci con 1 millón?

**3.6** [Ejercicio opcional] ¿Cómo se puede hacer para calcular el Fibonacci de valores grandes?

**3.7** ¿Qué concluyen sobre la complejidad de los problemas de *CodingBat Recursion 1* con respecto a los de *Recursion 2*?

#### 4) Simulacro de Parcial en el informe PDF



Para este simulacro, agreguen ***sus respuestas*** en el informe PDF.

Resuelva, como mínimo, los ejercicios marcados con **color rojo**.



***El día del Parcial no tendrán computador, JAVA o acceso a internet.***



**PISTA 1:** Véase ***Guía en Sección 4, Numeral 4.18*** “Respuestas del Quiz”

1. Pepito escribió un algoritmo que, dado un arreglo de enteros, decide si es posible escoger un subconjunto de esos enteros, de tal forma que la suma de los elementos de ese subconjunto sea igual a `target`. El parámetro `start` funciona como un contador y representa un índice en el arreglo de números `nums`.

```
01 public boolean SumaGrupo(int start,
                             int[] nums, int target) {
02     if (start >= nums.length) return target == 0;
03     return SumaGrupo(start + 1, nums,
                        target - nums[start])
04         || SumaGrupo(____, ____, ____);
05 }
```

**DOCENTE MAURICIO TORO BERMÚDEZ**

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)

¿Qué parámetros colocarían en el llamado recursivo de la línea 4 para que el programa funcione?

\_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_

**2.** Pepito escribió el siguiente código usando recursión

```
private int b(int[ ] a, int x, int low, int high) {  
    if (low > high) return -1;  
    int mid = (low + high)/2;  
    if (a[mid] == x) return mid;  
    else if (a[mid] < x)  
        return b(a, x, mid+1, high);  
    else  
        return b(a, x, low, mid-1);  
}
```

¿Cuál ecuación de recurrencia describe el comportamiento del algoritmo anterior para el peor de los casos?

- a)  $T(n)=T(n/2)+C$
- b)  $T(n)=2.T(n/2)+C$
- c)  $T(n)=2.T(n/2)+Cn$
- d)  $T(n)=T(n-1)+C$

**3.** Dayla y Kefo están aquí de nuevo. En esta vez han traído un juego muy interesante, en el cual Kefo, en primer lugar, escoge un numero  $n$  ( $1 \leq n \leq 20$ ) y, en segundo lugar, escoge tres números  $a, b$  y  $c$  ( $1 \leq a \leq 9, 1 \leq b \leq 9, 1 \leq c \leq 9$ ).

Después, Kefo le entrega estos números a Dayla y Dayla le tiene que **decir a Kefo la cantidad máxima de números tomados de  $a, b$  y  $c$  (se puede tomar un número más de una vez) que al sumarlos dan el valor  $n$ .**

Como un ejemplo, si Kefo escoge  $n=14$  y  $a=3, b=2, c=7$ . ¿Qué posibilidades hay de obtener 14 con  $a, b$  y  $c$ ?



7+7=14	cantidad es 2
7+3+2+2=14	cantidad es 4
3+3+3+3+2=14	cantidad es 5
...	
2+2+2+2+2+2+2=14	cantidad es 7

La cantidad máxima de números es 7. Esta sería la respuesta que da Dayla a Kefo. Como Dayla es muy astuta, ha diseñado un algoritmo para determinar la cantidad máxima de números y quiere que le ayudes a terminar su código.

Asuman que hay al menos una forma de sumar  $n$  usando los números  $a$ ,  $b$  y  $c$  en diferentes cantidades, incluso si algunos de los números se suman 0 veces como sucede en el ejemplo anterior.

```
1 int solucionar (int n, int a, int b, int c) {  
2     if (n == 0 || (n < a && n < b && n < c))  
3         return 0;  
4     int res = solucionar(_____) + 1;  
5     res = Math.max(_____, _____);  
6     res = Math.max(_____, _____);  
7     return res; }
```

### 3.1 Completen el espacio de la línea 04

\_\_\_\_\_

### 3.2 Completen los espacios de la línea 05

\_\_\_\_\_, \_\_\_\_\_

### 3.3 Completen los espacios de la línea 06

\_\_\_\_\_, \_\_\_\_\_

4. ¿Qué calcula el algoritmo desconocido y cuál es la complejidad asintótica en el peor de los casos del algoritmo desconocido?

```
01 public int desconocido(int[] a){  
02     return aux(a, a.length-1); }  
03  
04 public int aux(int[] a, int n){  
05     if(n < 1) return a[n];  
06     else return a[n] + aux(a, n-1); }
```

- a) La suma de los elementos del arreglo  $a$  y es
- b) Ordena el arreglo  $a$  y es  $O(n \log n)$
- c) La suma de los elementos del arreglo  $a$  y es  $O(1)$
- d) El máximo valor de un arreglo  $a$  y es  $O(n)$
- e) La suma de los elementos del arreglo  $a$  y es  $O(n)$

5. Hay un tablero de  $2 \times n$  cuadrados y usted necesita saber de cuantas maneras se puede llenar el tablero usando rectángulos de  $1 \times 2$ . Se ha propuesto el siguiente algoritmo recursivo

```
1 public int formas(int n){  
2     if(n <= 2) _____;  
3     return formas(_____) +  
4         formas(_____) ;  
5 }
```

5.1 Completen las líneas faltantes.

Línea 2. \_\_\_\_\_

Línea 3. \_\_\_\_\_

Línea 4. \_\_\_\_\_

5.2 ¿Cuántas instrucciones ejecuta el algoritmo en el peor de los casos?

- a.  $T(n) = T(n-1) + C$
- b.  $T(n) = T(n-1) + T(n-2) + C$
- c.  $T(n) = T(n/2) + C$
- d.  $T(n) = T(n+1) + C$

6. Alek y Krish están jugando *Número*. *Número* es un juego en el que un jugador 1, entrega un número  $n$  ( $1 \leq n \leq 10100$ ) a un jugador 2 y el jugador 2 debe determinar la suma de todos los dígitos de  $n$ , exceptuando el caso en el que hay dos dígitos adyacentes (es decir, contiguos, seguidos) que son iguales.

Si hay dos dígitos adyacentes, no se suma ninguno de los dos números adyacentes. Entre Alek y Krish escribieron un código para hacer esto más rápido, pero se ha borrado una parte. ¿Podrían ayudarles a reconstruir el código a Alek y Krish?

```
1 public int suma(String n) {
2     return sumaAux(n, 0);
3 }
4
5 private int sumaAux(String n, int i){
6     if (i >= n.length()) {
7         return 0;
8     }
9     if(i + 1 < n.length() &&
10        n.charAt(i) == n.charAt(i + 1)){
11         return _____;
12     }
13     return (n.charAt(i) - '0') + _____;
14 }
```

La operación `n.charAt(i) - '0'` convierte un caracter en su equivalente en entero, por ejemplo, el caracter `'1'` lo transforma en el número 1.

**6.1** Completen la línea 10.

\_\_\_\_\_

**6.2** Completen la línea 12.

\_\_\_\_\_

**7)** Dado un conjunto  $S$  de  $n$  elementos se quiere determinar si existe un subconjunto  $R$  de  $S$  tal que la suma de los elementos de  $R$  es igual a  $t$ , con la condición que, si se toma un elemento par, el siguiente elemento no puede estar en  $R$ .

```
1 public boolean comb(int[] S, int i, int t){
2     if(i >= S.length){
3         return t == 0;
4     }
5     //par
6     if(S[i] % 2 == 0){
7         return comb(S, i + 2, t - S[i])
8     }
9     return comb(S, i + 1, t);
10 }
```

```
8      || comb(S, i + 1, t); }  
9      return comb(_____) ||  
10     comb(_____) ;  
11 }
```

Al anterior código le faltan algunas líneas las cuales deben completar

### 7.1 Línea 9

\_\_\_\_\_

### 7.2 Línea 10

\_\_\_\_\_

**8)** El pequeño Polka fue a la tienda a comprar algunas cosas para su madre. El pequeño Polka pagó cierta cantidad de dinero y el tendero debe devolverle exactamente  $K$  pesos. El tendero tiene exactamente  $n$  diferentes tipos de monedas en su tienda, cada una de ellas con un valor  $v_i$ . Como el tendero no es muy astuto en asuntos de matemáticas, te ha pedido el favor de decirle de cuantas maneras puede él devolverle tal cantidad a Polka. Por favor ayúdanos a completar el siguiente código.

**Ejemplo:** Sea  $n = |v| = 3$ ,  $v = \{3, 4, 1\}$ ,  $K=7$ .

Las formas posibles de devolver 7 dólares serían 5:

1.  $1 + 1 + 1 + 1 + 1 + 1 + 1$
2.  $3 + 1 + 1 + 1$
3.  $3 + 3 + 1$
4.  $3 + 4$
5.  $4 + 1 + 1 + 1$

```
01 int cuantas(int K, int[] v, int n){  
02     if(K == 0){  
03         return 1;  
04     }  
05     boolean imposible;  
06     imposible = n <= 0 && K >= 1;  
07     imposible = imposible || K < 0;  
08     if(imposible){  
09         _____  
10     }  
11     int ni = cuantas(K, v, n - 1);  
12     int nj = cuantas(K - v[n-1], v, n);
```

```
13 int suma = _____;  
14 return suma;  
15 }
```

8.1. Complete Línea 9 .....

8.2. Complete Línea 13 .....

## 5. [Ejercicio Opcional] Lectura recomendada



"Quienes se preparan para el ejercicio de una profesión requieren la adquisición de competencias que necesariamente se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades..."

Tomado de <http://bit.ly/2qJKzJD>



Véase Guía en **Sección 3, numeral 3.5 y 4.20** de la Guía Metodológica, "Lectura recomendada" y "Ejemplo para realización de actividades de las Lecturas Recomendadas", respectivamente

Posterior a la lectura del texto **"Narasimha Karumanchi, Data Structures and Algorithms made easy in Java, (2<sup>nd</sup> edition), 2011. Chapter 3: Recursion and Backtracking"** realicen las siguientes actividades que les permitirán sumar puntos adicionales:

- a) Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras
- b) Hagan un mapa conceptual que destaque los principales elementos teóricos.



**NOTA :** Estas respuestas también deben incluirlas en el informe PDF

## 6. [Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual



El trabajo en equipo es una exigencia actual del mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques y trabajes bien con otras personas"

Tomado de <http://bit.ly/1B6hUDp>



Véase Guía en **Sección 3, numeral 3.6 y Sección 4, numerales 4.21, 4.22 y 4.23** de la Guía Metodológica

- a) Entreguen copia de todas las actas de reunión usando el *tablero Kanban*, con fecha, hora e integrantes que participaron
- b) Entreguen el reporte de *git* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron
- c) Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares



**NOTA:** Estas respuestas también deben incluirlas en el informe PDF

## 7. [Ejercicio Opcional] Laboratorio en inglés:



El inglés es un idioma muy importante en la Ingeniería de Sistemas porque la mayoría de los avances en tecnología se publican en este idioma y la traducción, usualmente se demora un tiempo y es sólo un resumen de la información original.

Adicionalmente, dominar el inglés permite conseguir trabajos en el exterior que son muy bien remunerados

Tomado de [goo.gl/4s3LmZ](https://goo.gl/4s3LmZ)

Entreguen el código y el informe traducido al inglés. Utilicen la plantilla dispuesta en este idioma para el laboratorio.

## Resumen de ejercicios a resolver (sin opcionales)


**1.1** Escriba una función que reciba como parámetro un valor entero  $n$  y retorne el  $n$ ésimo término de la serie de Fibonacci.

**2.1** Resolver al menos 5 ejercicios del nivel Recursion 1 de CodingBat:  
<http://codingbat.com/java/Recursion-1>

**2.2** Resolver al menos 5 ejercicios del nivel Recursion 2 de la página CodingBat:  
<http://codingbat.com/java/Recursion-2>

**3.1** Expliquen con sus propias palabras cómo funciona el ejercicio GroupSum5

**3.2** Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y 2.2 y agréguelas al informe PDF

	<b>UNIVERSIDAD EAFIT</b> <b>ESCUELA DE INGENIERÍA</b> <b>DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</b>	<b>Cód. ST0245</b>
		<b>Estructuras de Datos 1</b>

**3.3** Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral anterior

**3.4** ¿Qué aprendieron sobre Stack Overflow?

**3.5**Cuál es el valor más grande que pudo calcular para Fibonacci? ¿Por qué? ¿Por qué no se puede ejecutar Fibonacci de forma recursiva con 1 millón?

**3.7** ¿Qué concluyen sobre la complejidad de los problemas de CodingBat Recursion 1 con respecto a los de Recursion 2?

**4.** Simulacro de Parcial



## **Ayudas para resolver los ejercicios**

<b>Ayudas para el Ejercicio 1.1.....</b>	<b><u><a href="#">Pág. 16</a></u></b>
<b>Ayudas para el Ejercicio 2.1.....</b>	<b><u><a href="#">Pág. 16</a></u></b>
<b>Ayudas para el Ejercicio 2.2.....</b>	<b><u><a href="#">Pág. 17</a></u></b>
<b>Ayudas para el Ejercicio 2.4.....</b>	<b><u><a href="#">Pág. 17</a></u></b>
<b>Ayudas para el Ejercicio 3.2.....</b>	<b><u><a href="#">Pág. 18</a></u></b>
<b>Ayudas para el Ejercicio 3.4.....</b>	<b><u><a href="#">Pág. 18</a></u></b>
<b>Ayudas para el Ejercicio 3.5.....</b>	<b><u><a href="#">Pág. 19</a></u></b>
<b>Ayudas para el Ejercicio 3.6.....</b>	<b><u><a href="#">Pág. 19</a></u></b>
<b>Ayudas para el Ejercicio 4.....</b>	<b><u><a href="#">Pág. 19</a></u></b>
<b>Ayudas para el Ejercicio 5a.....</b>	<b><u><a href="#">Pág. 19</a></u></b>
<b>Ayudas para el Ejercicio 5b.....</b>	<b><u><a href="#">Pág. 20</a></u></b>
<b>Ayudas para el Ejercicio 6a.....</b>	<b><u><a href="#">Pág. 20</a></u></b>
<b>Ayudas para el Ejercicio 6b.....</b>	<b><u><a href="#">Pág. 20</a></u></b>
<b>Ayudas para el Ejercicio 6c.....</b>	<b><u><a href="#">Pág. 20</a></u></b>

## Ayudas para el Ejercicio 1.1



**PISTA 1:** Véase **Guía sección 4, numeral 4.6** “Cómo usar la escala logarítmica en Microsoft Excel”.



**PISTA 2:** Véase **Guía sección 4, numeral 4.4** “Cómo aumentar el tamaño del heap y del stack en Java”



**PISTA 3:** Véase **Guía sección 4, numeral 4.5** “Cómo visualizar el montículo (heap) y el stack, y el consumo total de memoria de Java”

## Ayudas para el Ejercicio 2.1

### Error común



## Ayudas para el Ejercicio 2.2



**PISTA 1:** El algoritmo *GroupSum* falla porque al llamarse recursivamente con el parámetro `start` se queda en una recursión infinita



**PISTA 2:** El algoritmo *GroupSum* falla porque al llamarse recursivamente con el parámetro `start-1` se sale del arreglo cuando `start = 0`.



**PISTA 3:** El algoritmo *GroupSum* falla porque al llamarse recursivamente con el parámetro `start` se sale del arreglo cuando `start = length-1`



A continuación encontrarán algunos errores comunes que pueden ser aplicados con los ejercicios

```
public boolean groupSum(int start, int[] nums, int target) {  
    if (start >= nums.length) return target == 0;  
    return groupSum(start+1, nums, target - nums[start])  
        || groupSum(start, nums, target );  
}
```

```
public boolean groupSum(int start, int[] nums, int target) {  
    if (start >= nums.length) return target == 0;  
    return groupSum(start+1, nums, target - nums[start])  
        || groupSum(start-1, nums, target );  
}
```

```
public boolean groupSum(int start, int[] nums, int target) {  
    if (start >= nums.length) return target == 0;  
    return groupSum(start+1, nums, target - nums[start])  
        || groupSum(start+1, nums, target - nums[start - 1] );  
}
```

## Ayudas para el Ejercicio 2.5



**PISTA 1:** Usen un algoritmo para corroborar si es un grafo bipartito. Léase qué es bipartito en <http://bit.ly/2hGwAo2>

## Ayudas para el Ejercicio 2.6

**DOCENTE MAURICIO TORO BERMÚDEZ**  
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627  
Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)



**PISTA 1:** Algoritmos para hallar componentes fuertemente conexos. Ordenamiento topológico. DFS. Léase en <http://bit.ly/2gTeJKh>

## Ayudas para el Ejercicio 3.1

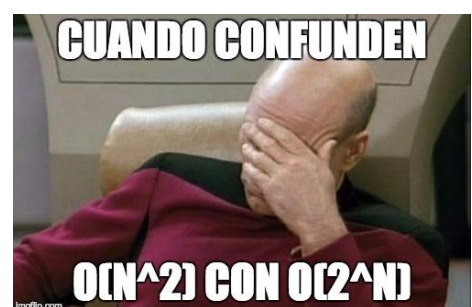
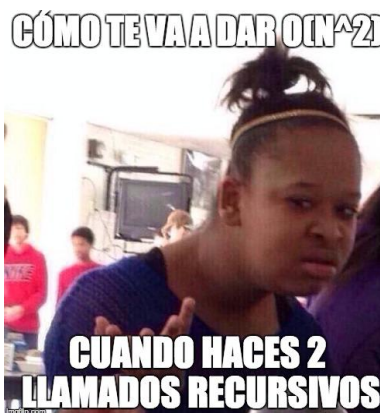


**PISTA:** Véase *Guía en Sección 4, numeral 4.11* “Cómo escribir la complejidad de un ejercicio en línea”



**PISTA 2:** Véase *Guía en Sección 4, numeral 4.19* “Ejemplos para calcular la complejidad de un ejercicio de CodingBat”

### Errores Comunes



## Ayudas para el Ejercicio 3.2



**PISTA 1:** Véase *Guía sección 4, numeral 4.6* “Cómo usar la escala logarítmica en Microsoft Excel”.

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)



**PISTA 2:** Si todos los tiempos de un algoritmo dan más de 5 minutos realice otra tabla, para ese algoritmo, tomando tiempos para arreglos de tamaño 1000, 10000 y 100000.

## Ayudas para el Ejercicio 3.4



**PISTA 1:** <http://bit.ly/1TRr3HL>

## Ayudas para el Ejercicio 3.5



**PISTA 1:** <http://bit.ly/2ix7rjl>

## Ayudas para el Ejercicio 3.6



**PISTA 1:** <http://bit.ly/2ix7rjl>

## Ayudas para el Ejercicio 4



**PISTA 1:** Lean las diapositivas tituladas “*Data Structures I: O Notation (recursion)*” encontrarán la mayoría de las respuestas



**PISTA NUMERAL 5.2:** Consideren llenar un tablero de  $2 \times n$ . Si le quitamos la primera baldosa tenemos un tablero de  $2 \times (n-1)$  (usando recursión). Si le quitamos 2 baldosas queda un tablero de  $2 \times (n-2)$ . Hagan el dibujo. ¿Se le pueden quitar 3 baldosas, o con lo anterior ya puedo formar el de  $2 \times (n-3)$ ?

## Ayudas para el Ejercicio 5a



**PISTA 1:** En el siguiente enlace, unos consejos de cómo hacer un buen resumen <http://bit.ly/2knU3Pv>



**PISTA 2:** [Aquí](#) le explican cómo contar el número de palabras en Microsoft Word

## Ayudas para el Ejercicio 5b



**PISTA 1:** Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en <https://cacoo.com/> o <https://www.mindmup.com/#m:new-a-1437527273469>

## Ayudas para el Ejercicio 6a



**PISTA 1:** Véase **Guía en Sección 4, Numeral 4.21** “Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban”

## Ayudas para el Ejercicio 6b



**PISTA 1:** Véase **Guía en Sección 4, Numeral 4.23** “Cómo generar el historial de cambios en el código de un repositorio que está en svn”

## Ayudas para el Ejercicio 6c



**PISTA 1:** Véase **Guía en Sección 4, Numeral 4.22** “Cómo ver el historial de revisión de un archivo en Google Docs”

